POCKET COMPUTER NEWSLETTER

© Copyright 1982 — Issue 20 December

EPSON AMERICA ANNOUNCES HX-20 PORTABLE COMPUTER

The company that has marketed the highly successful MX- series of low cost computer printers for several years has now announced a major entry in the burgeoning portable computer market. The new model is designated the HX-20. Its weight, including a built-in 24-column dot matrix microprinter is under four pounds. The unit also includes a full-sized keyboard containing 68 keys and a four-line by 20-character LCD (liquid-crystal display) that can also display simple graphics in a 120 by 32 dot matrix mode. Overall size of the unit if 11-3/8 by 8-1/2 by 1-3/4 inches. The area it occupies on a table is just about exactly that of an ordinary piece of writing paper.

Other features of the unit include: dual CMOS 8-bit type 6301 microprocessors operating at 614 KHz; 16K of RAM provided with the basic unit (expandable to 32K); 32K of system ROM supplied as standard (expandable to 40K internally, 64K in an expanded mode); built-in timer/clock/calendar with separate battery backup; four octave, half-tone, programmable tone generator; RS-232C communications port at full or half duplex and 110 to 4800 baud rate; built-in serial interface; ability to interface with HP barcode reader (through a special connector); standard audio cassette interface; optional microcassette unit can be installed in the basic unit; rechargeable Nickel-Cadmium batteries can power the unit for up to 50 hours of completely portable operation and recharge within 8 hours.

The company is reportedly making initial deliveries to selected, authorized dealers. The list price for the basic unit is $799.95. For additional information contact: **Epson America, Inc.**, *3415 Kashiwa Street, Torrance, CA 90505.* The phone number is (213) 539-9140.

Photo *Epson HX-20 Portable Computer*

Another *Personal Information*™ product from SCELBI Publications.

A WEATHER FORECASTING PROGRAM

The accompanying program predicts the local, short term (12 – 24 hours) weather for any region in the Northern Hemisphere above the Tropic of Cancer.

To use the program, obtain the barometric pressure reading (inches). Try to interpolate readings to the nearest hundredth of an inch, An example reading might be: 29.95.

You must observe barometer movements over several hours to determine the barometric tendency. For application of this program, the following definitions apply: The barometric tendency is moving fast if it is changing more than 0.05 inches per hour. It is slow if the change is between 0.02 and 0.05 inches per hour. It is considered steady if less than or equal to 0.02 inches per hour.

Enter wind direction as N, NE, E, SE, S, SW, W or NW depending on which direction the wind is blowing *from* (not the direction that it is going towards!).

The program is based on meteorological rules of thumb. To obtain highest possible accuracy, the program should be run at frequent intervals as atmospheric conditions change.

The listing shown is for a PC-2/PC-1500. The program will fit in a PC-1/PC-1211 but you will need to change the variable @$(Z) in line 210 to read A$(Z). This program submitted by: *Emerich Auersbacher, 41 King Street – Apt. 2, Belleville, NJ 07109.*

```
10:BEEP 1:PAUSE "
   WEATHER FORECA
   STER"
20:A$="S":B$="SW"
   :C$="W":D$="NW
   ":E$="N":F$="N
   E":G$="E":H$="
   SE"
30:INPUT "BAROMET
   ER (IN):";P:IF
   P>30.2LET S=0
40:IF P<30.2LET S
   =1
50:IF P<30.1LET S
   =2
60:IF P<30.0LET S
   =3
70:IF P<=29.8LET
   S=4
80:INPUT "(R)ISE,
   (F)ALL, (S)TEAD
   Y?";T$:X=0
90:IF T$="R"INPUT
   "RISING (F)AST
   ,(S)LOW?";R$:
   IF R$="F"LET X
   =1:GOTO 150
100:IF T$="R"LET X
   =2:GOTO 150
110:IF T$="F"INPUT
   "FALLING (F)AS
   T,(S)LOW?";R$:
   IF R$="F"LET X
   =3:GOTO 150
120:IF T$="F"LET X
   =4:GOTO 150
130:IF T$="S"GOTO
   150
140:GOTO 80
150:INPUT "WIND FR
   OM THE:";W$:
   GOSUB 200
160:IF Y=0BEEP 3:
   PAUSE "ERROR":
   GOTO 150
170:GOTO 300
200:Y=0:FOR Z=1TO
   8
210:IF W$=@$(Z)LET
   W=Z:Z=8:Y=1:
   GOTO 220
220:NEXT Z:RETURN
300:T=600:Y=ABS (W
   -3)<2
310:IF Y*(S<3)*(X=
   0)LET Z=1:GOTO
   T
320:IF Y*(S=1)*(X=
   1)LET Z=6:GOTO
   T
330:IF Y*(S=0)*(X=
   4)LET Z=3:GOTO
   T
340:Y=(W=1)+(W=8)
350:IF Y*(S=1)*(X=
   4)LET Z=4:GOTO
   T
360:IF Y*(S=1)*(X=
   3)LET Z=5:GOTO
   T
370:Y=ABS (W-7)<2
380:IF Y*(S=1)*(X=
   4)LET Z=4:GOTO
   T
390:IF Y*(S=1)*(X=
   3)LET Z=5:GOTO
   T
400:IF Y*(S>2)*(X=
   4)LET Z=6:GOTO
   T
410:IF Y*(S>2)*(X=
   3)LET Z=7:GOTO
   T
420:Y=(W=6)+(W=7)
430:IF Y*(S<2)*(X=
   4)LET Z=8:GOTO
   T
440:IF Y*(S<2)*(X=
   3)LET Z=9:GOTO
   T
450:IF ((W=1)+(W=2
   ))*(S>2)*(X=2)
   LET Z=10:GOTO
   T
460:IF ((W<2)+(W>6
   ))*(S=4)*(X=3)
   LET Z=11:GOTO
   T
470:IF (ABS (W-6)<
   2)*(S=4)*(X=3)
   LET Z=12:GOTO
   T
480:IF (ABS (W-3)<
   2)*(S=4)*(X=1)
   LET Z=13:GOTO
   T
490:IF (X<3)*(ABS
   (W-5)<3)LET Z=
   2:GOTO T
500:Z=1:IF Z=3LET
   Z=8
600:BEEP 3:PAUSE "
    FORECAST -->"
   :GOSUB T+10*Z:
   GOTO 30
610:PRINT "FAIR, L
   ITTLE CHANGE."
   :RETURN
620:PRINT "FAIR, A
    LITTLE COOLER
   .":RETURN
630:PRINT "FAIR, A
    LITTLE WARMER
   .":RETURN
640:PRINT "RAIN WI
   THIN 24 HRS.":
   RETURN
650:PRINT "WINDY,
   RAIN BY 12 HRS
   .":RETURN
660:PRINT "INCREAS
   ING CLOUDS, RA
   IN.":RETURN
670:PRINT "RAIN AN
   D HIGH WINDS."
   :RETURN
680:PRINT "UNSETTL
   ED, CHANCE RAI
   N.":RETURN
690:PRINT "RAIN/SN
   OW, WINDY.":
   RETURN
700:PRINT "CLEARIN
   G, BECOMING FA
   IR.":RETURN
710:PRINT "SEVERE
   STORM IMMINENT
   .":RETURN
720:PRINT "HEAVY R
   AIN/SNOW, WIND
   S.":RET
730:PRINT "CLEARIN
   G AND COOLER."
   :RETURN
STATUS 1        1608
```

PRODUCT REVIEW
99 TIPS & TRICKS FOR THE NEW POCKET COMPUTERS

Author: *Jim Cole*
Publisher: *ARCsoft Publishers, POB 132, Woodsboro, MD 21798*
List Price: *$7.95 (add $1 for s/h charges on mail orders).*
Availability: *Directly from publisher, computer & book stores.*
Reviewer: *Nat Wadsworth, Editor.*

At first I thought "maybe old Jim just re-arranged some of his previous routines from his earlier books?" But no (how could I think such a thing?), Jim has indeed come up with a genuine potpourri of new ideas and original programs for the Radio Shack PC-2/Sharp PC-1500 Pocket Computers!

These routines range from the almost ridiculously simple (such as laying down a string of "#" signs to portray railroad tracks!) all the way up to complete programs that allow you to do such things as simulate a RPN calculator, make an automatic Morse Code practice oscillator (really neat!) and convert world currencies at the touch of a few buttons. There are also quite a few graphics oriented routines, including one that draws a pair of little dice on the LCD and makes them appear just as though you had rolled 'em!

While this 128 page booklet is intended primarily for beginners, it has enough interesting ideas so that even those with plenty of experience might pick up a trick or two. The price makes it a bargain.

GETTING STARTED WITH MACHINE LANGUAGE

Perhaps some of you who are wondering what all the fuss over the machine codes for the PC-1500/PC-2 is about, might be able to satisfy your curiosities somewhat by trying your hand at a few simple programs written in machine language.

Let's start with a program so simple that it is of strictly of educational value. The program to be illustrated first will merely perform the equivalent of the BASIC statement C = A + B, and that is all! In fact, it will use routines that are already stored in ROM for the purpose of performing floating-point arithmetic.

Three things need to be done: 1) Write the program; 2) Put the program into memory; and 3) Execute the machine-language program.

We will begin by writing the program. First we need to transfer the contents of the variable named A into the floating-point accumulator, which is CPU register R0. The ROM routine for doing this requires that CPU register YH contain the code &7A. Furthermore, CPU register X must contain the address at which variable A is stored. Variable A is stored starting at address &7900. Here is what the start of the program looks like:

Opcodes	Mnemonics	Comments
58 7A	LDYH #7A	Contents of variable A
48 79	LDXH #79	placed into register R0
4A 00	LDXL #00	
BE F7 3F	JSR F73F	

Next, we will move the contents of register R0 into register R2. This needs to be done so that variable B may be placed into R0;

E6	CALL E6	Stores R0 into R2

Remember that the opcode E6 used alone is equivalent to the instruction CALL E6. (We could have used opcodes CD E6 for this directive.)

Now we need to put variable B into R0. (Since the previous instruction has left &7A in CPU register YH, we do not need to include the LDYH #7A directive this time.)

Opcodes	Mnemonics	Comments
48 79	LDXH #79	Contents of variable B
4A 08	LDXL #08	stored into R0
BE F7 3F	JSR F73F	

The multiplication itself can be performed by simply using the ROM routine for floating-point multiplication:

CD 7E	CALL 7E	Replace R0 with R0 * R2

Finally, the calculated product must be placed into variable C, whose beginning address is &7910:

58 79	LDYH #79	Contents of R0 put into
5A 10	LDYL #10	variable C
BE F7 11	JSR F711	

We must end the machine-language program with:

9A	RTS	Return from subroutine (to BASIC)

That is the program! Now, how do we load it into an appropriate section of memory?

First, we need to reserve some space for the machine-language routine. We can do this be executing NEW &3900. (If you are not using an 8K CE-155 module then use NEW &4100 instead.) This action protects the memory area from &38C5 to &3900 (or &40C5 to $4100) from being used by a BASIC program.

The opcodes derived above must now be put into memory using a POKE statement. This needs to be done just once. It may be accomplished either in manual mode or in a program line. The following statements will perform this operation:

POKE &38D0, &58, &7A, &48, &79, &4A, 0, &BE, &F7, &3F, &E6, &48, &79, &4A, 8, &BE, &F7

POKE &38E0, &3F, &CD, &7E, &58, &79, &5A, &10, &BE, &F7, &11, &9A

If you are not using an 8K expansion module, then begin the two POKE sequences at &40D0 and &40E0 respectively.

Now you are in a position to test the program. Put a couple of your favorite numbers into variables A and B. Then, either in manual mode or in a program line, execute the BASIC statement: CALL &38D0. (Use CALL &40D0 if you do not have an 8K module). This will cause the machine language program that you stored at &38D0 (or &40D0) to be executed. To see whether things worked as expected, display variable C. It should contain the product of A and B.

If errors are made in a machine-language routine, there are no "error messages" provided by the PC to assist the programmer. Programming errors in machine-language frequently lead to a "crash," disabling the keyboard or disrupting operations in other ways. In such cases, you may be forced to use the "All Reset" button on the back of the PC, then start all over again! Anyone who takes up experimenting with machine-language programming can expect this to happen often!

A More Useful Machine-Language Routine

The factorial function is not a part of the Sharp PC-1500/Radio Shack PC-2's repertoire. Of course, it can be programmed in BASIC and one efficient algorithm that will calculate N! (factorial) when N is an integer in the range 1 to 69 may be programmed as follows:

```
10 F=1: FOR J=N TO 2 STEP -1: F=F*J: NEXT J
```

The result is stored in variable F. The length of time required to perform this calculation using the BASIC routine depends on the value of N. When N is specified as 69, it takes approximately 2.4 seconds.

A machine-language program that accomplishes the same result is shown below. It will reduce the time required to calculate 69! to just 0.6 seconds. If this increase in speed does not seem that impressive, it should be remembered that there is a lot of arithmetic being done here and that the same floating-point routines are being used whether BASIC or machine language is used. All of the savings in time is being achieved from the simple elimination of the need to interpret the BASIC statement line!

There is even another advantage to using the machine-language routine for this type of calculation. In the BASIC version, each intermediate result is rounded to ten digits. This roundoff accumulates. The machine-language version maintains twelve digits throughout the computation, rounding to ten digits only after the final result is obtained. As an example, the calculated result for 56! errs by 8 in the tenth digit when using the BASIC program, but the machine-language program is correct to all ten digit positions.

The routine shown is accessed by a CALL instruction from BASIC in which the input, N, is passed to the machine-language program. This input is placed into CPU register X by the call routine. However, when this is done it is in hexadecimal format. It must be converted to decimal before the floating-point arithmetic routines may be applied.

Fortunately, a ML program for making this conversion already exists in ROM. The base page addressed subroutine 10 will store the BCD equivalent of the contents of register U in R0. (This subroutine requires the passing of one byte. The desired effect can be achieved by using &80.)

Here is the program. The memory addresses assume storage in a PC having an 8K RAM module.

Address	Opcodes	Mnemonics	Comments
&38D0	FD 6A	STX U	X stored into U
&38D2	FD A8	PSH U	Save in stack
&38D4	CD 10 80	CALL 10 (PASS 80)	Store into R0 (in BCD)
&38D7	E6	CALL E6	R0 into R2
&38D8	FD 2A	POP U	Get U from stack
&38DA	62	DEUL	Decrement UL
&38DB	6E 02	CPUL #02	Compare UL to 2
&38DD	81 09	FBNC #09	If UL<2, branch to &38E8
&38DF	FD A8	PSH U	Save U in stack
&38E1	CD 10 80	CALL 10 (PASS 80)	U stored into R0 (in BCD)
&38E4	CD 7E	CALL 7E	R0 replaced by R0 * R2
&38E6	9E 11	RB #11	Branch to &38D7
&38E8	BE F9 32	JSR F932	Round R0 to ten digits
&38EB	58 79	LDYH #79	R0 is copied
&38ED	5A 28	LDYL #28	into variable F,
&38EF	BE F7 11	JSR F711	which begins at &7928
&38F2	9A	RTS	Return to BASIC

The program may be loaded into memory using the POKE statements:

POKE &38D0, &FD, &6A, &FD, &A8, &CD, &10, &80, &E6, &FD,

&2A, &62, &6E, 2, &81, 9, &FD
POKE &38E0, &A8, &CD, &10, &80, &CD, &7E, &9E, &11, &BE, &F9, &32, &58, &79, &5A, &28, &BE
POKE &38F0, &F7, &11, &9A

Execution of CALL &38D), N will place the value of N! into the variable named F. Of course, the number (1 to 69) whose factorial is to be calculated must be assigned to the variable N beforehand.

As in the previous example, if you do not have an 8K memory expansion module, the addresses in the POKE statements should be changed to &40D0, &40E0 and &40F0 with CALL &40D0 being used to execute the routine.

One More Example

This final example program makes use of the floating-point stack. This stack should not be confused with the stack used by machine-language PUSH and POP instructions.

The ML (machine-language) instructions PSHA, PSHX, etc., will save one or two bytes obtained from a CPU register in a stack located in RAM at addresses &784F and below. The instructions POPA, POPX, etc., remove one or two bytes at a time from this stack and store them back into the corresponding CPU register(s). This same stack is used to save machine-language subroutine return addresses.

A completely separate stack, similar in principle, is created by ROM routines in the PC. This stack is used to save an 8-byte block of data, obtained from floating-point register R0. This data can later be popped back into R0. Just as in the case of the CPU stack, this software-created stack operates on a LIFO (last-in, first-out) basis. The floating-point stack is located between &7A38 and &7AFF. It is also used to save BASIC subroutine return addresses and in keeping track of FOR/NEXT operations.

The program shown next will perform calculations equivalent to the BASIC statement: C = $\sqrt{}$(A*A + B*B).

Address	Opcodes			Mnemonics	Comments
&38D0	58	7A		LDYH #7A	Contents of
&38D2	48	79		LDXH #79	variable A
&38D4	4A	00		LDXL #00	stored
&38D6	BE	F7	3F	JSR F73F	into R0
&38D9	BE	F0	19	JSR F019	Contents of R0 squared
&38DC	BE	DB	F5	JSR DBF5	R0 pushed onto F-P stack
&38E0	48	79		LDXH #79	Contents of
&38E2	4A	08		LDXL #08	variable B
&38E4	BE	F7	3F	JSR F73F	stored into R0
&38E7	BE	F0	19	JSR F019	Contents of R0 squared
&38EA	E6			CALL E6	Store R0 into R2
&38EB	CD	30		CALL 30	Pop from F-P stack into R0
&38ED	F0			CALL F0	Add R2 to R0
&38EF	BE	F0	E9	JSR F0E9	Replace R0 by its square root
&38F2	BE	F9	32	JSR F932	Round R0 to ten digits
&38F5	58	79		LDYH #79	Store contents
&38F7	5A	10		LDYL #10	of R0 into
&38F9	BE	F7	11	JSR F711	variable C
&38FC	9A			RTS	Return to BASIC

As in the previous examples, use POKE statements to load the opcodes into memory. Assign values to variables A and B then execute the routine by CALL &38D0 (or CALL &40D0 if you do not have an 8K module). The variable C will contain the computed result.

These examples provide a rough idea of how floating-point routines provided as part of the ROM can be incorporated into ML programs. Of course, there are many applications of machine language that would not involve the use of these floating-point ROM algorithms. Remember, essentially anything that can be done in BASIC can be done in ML, but the process can be a lot more complicated. The payback is in operating speed and increased flexibility.

[Sleuth **Norlin Rober** *provides a wealth of information regarding the location of ROM floating-point routines in the following tables. Just at press time he notified* **PCN** *that he had located one more key F-P entry point:* Calling &EFB6 will perform floating-point subtraction between F-P registers R0 and R2 (as R0 - R2) with the result left in R0. *That provides a pretty complete F-P set of ROM routines for you to work with. Hats off to Norlin once again! — N.W.]*

FLOATING-POINT ARITHMETIC IN THE PC-1500

Floating-point arithmetic is performed using 12-digit binary-coded-decimal mantissas, with exponents of 10 stored in two's-complement binary. Each floating-point number is stored in 8 bytes, as follows:

BYTE	CONTENTS (NORMALIZED FORM)
Ø	Exponent, in two's-complement binary
1	Sign of mantissa; Ø for positive, &80 for negative
2	1st and 2nd digits of mantissa, in BCD
3	3rd and 4th digits
4	5th and 6th digits
5	7th and 8th digits
6	9th and 10th digits
7	11th and 12th digits

BASIC routines round calculated results to ten-digit mantissas prior to display, printing, or storage into a variable.

Computations are carried out using seven floating-point registers, each comprised of 8 bytes, located in RAM at the following addresses:

RØ: 7A00 to 7A07 (Floating-point Accumulator)
R1: 7A08 to 7A0F
R2: 7A10 to 7A17
R3: 7A18 to 7A1F
R4: 7A20 to 7A27
R5: 7A28 to 7A2F
R6: 7A30 to 7A37

In ROM routines, CPU Registers X and Y act as pointers to the addresses used as floating-point registers.

In the major routines, calculated results are placed in RØ. In binary operations, the operands taken as input arguments are obtained from RØ and R2; operations using a single operand use the contents of RØ as input.

MAJOR SUBROUTINES INVOLVING FLOATING-POINT ARITHMETIC

The ROM subroutines tabulated below are useful in machine-language programming where floating-point arithmetic is used.

Where applicable, base-page call numbers are tabulated also.

CALL	ADDR	RESULT IN RØ	REGISTERS USED	NOTES
FØ	EFBA	RØ + R2	RØ, R2	
7E	F01A	RØ • R2	RØ to R5	
58	F084	RØ / R2	RØ to R4	
	F019	RØ • RØ	RØ to R5	
6E	F080	1 / RØ	RØ to R4	XH and YH must contain &7A
	F0E9	SQR RØ	RØ to R2	
	F161	LN RØ	RØ to R5	
	F165	LOG RØ	RØ to R5	
	F1CB	EXP RØ	RØ to R5	
	F1D4	10 ^ RØ	RØ to R5	
	F391	COS RØ	RØ to R6	
	F39E	TAN RØ	RØ to R5	
	F3A2	SIN RØ	RØ to R6	
	F492	ACS RØ	RØ to R6	
	F496	ATN RØ	RØ to R5	Flag C must be clear
	F49A	ASN RØ	RØ to R6	
	F531	DEG RØ	RØ to R4, R6	
	F564	DMS RØ	RØ to R6	
	F597	ABS RØ	RØ	
	F59D	SGN RØ	RØ	
	F5B5	π	RØ, R2	π placed in R2 also
	F5BE	INT RØ	RØ, R2	R2 = fractional part (not normalized)
	F5DD	RND RØ	RØ to R6	
EC	F757	zero	RØ	
	F89C	RØ ^ R2	RØ to R6	Calculated using logarithms
	F932	RØ, rounded	RØ	Rounded to ten significant digits

Trigonometric functions and their inverses are calculated in degrees, radians, or grads, according to the mode set.

Except as noted, all results are normalized. A result less in absolute value than 1 E-99 is replaced by zero.

Illegal operations end the subroutine with Flag C set; CPU register UH will contain the error code, as listed below:

UH = &25: ERROR 37, overflow
UH = &26: ERROR 38, division by zero
UH = &27 ERROR 39, illegal function argument

ROUTINES FOR TRANSFERRING FLOATING-POINT DATA

A Floating-point stack shares RAM addresses 7A38 to 7AFF with the BASIC loop and subroutine stacks.

The ROM subroutines listed below include data transfers involving the floating-point registers, floating-point stack, and numeric variables.

CALL	ADDR	EFFECT OF SUBROUTINE	NOTES
80	F707	RØ stored into R6	
B6	F70D	RØ stored into R2	
	F737	R6 stored into RØ	
56	F73D	R2 stored into RØ	XH and YH must contain &7A
64	F7B5	RØ and R6 exchanged	XH and YH must contain &7A
66	F7B9	RØ and R2 exchanged	XH and YH must contain &7A
	DBF5	Push RØ onto FP stack	If capacity exceeded, Flag C set and UH = &ØE
30	DC16	Pop RØ from FP stack	
	F711	RØ stored into variable	XH must contain &7A, Y the variable address
	F73F	Variable stored into RØ	YH must contain &7A, X the variable address

ADDRESSES OF FIXED VARIABLES A TO Z

The addresses given below are those of the initial bytes of the respective variables. They are the addresses that must be in CPU register X or Y, in the above routines for data transfer between RØ and variables.

A 7900	G 7930	N 7968	T 7998
B 7908	H 7938	O 7970	U 79A0
C 7910	I 7940	P 7978	V 79A8
D 7918	J 7948	Q 7980	W 79B0
E 7920	K 7950	R 7988	X 79B8
F 7928	L 7958	S 7990	Y 79C0
	M 7960		Z 79C8

NORMALIZING ROUTINES

The subroutines for transforming calculated results into normalized form are executed by the major floating-point routines.

CALL	ADDR	EFFECT	NOTES
	F65D	RØ normalized	XH and YH must contain &7A
E8	F661	RØ normalized and given a positive sign	XH and YH must contain &7A
52	F663	RØ normalized; contents of A used as sign	XH and YH must contain &7A

SHORT SUBROUTINES USED WITH FLOATING-POINT ARITHMETIC

In certain subroutines (as indicated) CPU Registers XH and/or YH must contain 7A prior to execution of the subroutine.

Unless otherwise stated, the initial byte of a floating-point register, representing the exponent, is ignored.

CALL	ADDR	7A REQ'D	EFFECT OF SUBROUTINE
	F63C		11th and 12th digits of RØ cleared
7C	F6B6	XH, YH	Sign of R2 and sign of product RØ*R2 PUSHed; both signs cleared
6C	F6FB	XH	Sign of RØ into A; sign cleared
	F701	XH, YH	R2 copied into R6, including exponent
80	F707		RØ copied into R6, including exponent
B6	F70D		RØ copied into R2, including exponent
68	F715	XH, YH	R6 copied into R2, including exponent
	F71B	XH, YH	R2 copied into RØ
	F71F	XH, YH	R1 copied into RØ
	F725	XH, YH	R2 copied into R1
82	F729	XH, YH	RØ copied into R2
78	F72F	XH, YH	RØ copied into R1
	F733		Addresses X to X+6 copied into Y to Y+6
	F737		R6 copied into RØ, including exponent
56	F73D	XH, YH	R2 copied into RØ, including exponent
70	F747	XH	R2 cleared
	F74B	XH	R1 cleared
	F74F	XH	R5 cleared
	F753	XH	R2 cleared, including exponent
EC	F757		RØ cleared, including exponent
	F75B		Addresses X to X+7 cleared
76	F75F	XH	RØ cleared
	F761		Addresses X to X+6 cleared
BA	F763		Addresses X to X+UL cleared
	F769	XH	Digits of R5 shifted right
	F76D	XH	Digits of R1 shifted right
	F771	XH	Digits of R2 shifted right
74	F775	XH	Digits of RØ shifted right
	F777		Digits of addresses X to X+6 shifted right
	F780	XH	Bytes of R1 shifted right
	F78C		Digits of bytes 2 to 6 of R6 shifted left
	F794	XH	Digits of R5 shifted left
	F798	XH	Digits of R1 shifted left
EA	F79C	XH	Digits of RØ shifted left
5E	F7A7	YH	Random number copied into RØ
54	F7BØ		XH and YH loaded with #7A
64	F7B5	XH, YH	RØ and R6 are exchanged, including exponent
66	F7B9	XH, YH	RØ and R2 are exchanged, including exponent
	F7C8	XH, YH	R3 is added to RØ
EE	F7CC	XH, YH	R2 is added to RØ
72	F7CE	XH	Register that ends with address Y is added to RØ
	F7D9	XH, YH	R4 is subtracted from RØ
7A	F7DD	XH, YH	R2 is subtracted from RØ
	F7DF	XH	Register that ends with address Y is subtracted from RØ
	F7E1		Addresses Y-6 to Y subtracted from addresses X-6 to X
	F7EB	XH	RØ is replaced by its ten's complement

SHORT SUBROUTINES USED WITH FLOATING-POINT ARITHMETIC (CONT'D)

CALL	ADDR	7A REQ'D	EFFECT OF SUBROUTINE
	F7F7		Digits of Register ending with X shifted right A times
	F805		Bytes of Register ending with X shifted right A times
	F81C	XH, YH	2*R2 into R3, 4*R2 into R4
	F820	XH, YH	2*R2 into R3, 4*R2 into R4, 8*R2 into R5
	F83C	XH, YH	'LDA 7A08. If A≠Ø digits of R1 are shifted right A-1 times, and R1 is added to R2
	F84D	XH, YH	LDA 7A08. If A≠Ø, bytes of R1 are shifted right A-1 times, and R1 is added to R2
	F85A	YH	If GRAD Mode, R2 = 0.9; if RADIAN, R2 = 57.2957795131
	F866	YH	R2 set equal to 57.2957795131 (π/180)
	F875	YH	R2 set equal to 3.14159265359 (π)
	F87B	YH	R2 set equal to 0.434294481903 (LOG e)
	F87F	YH	R2 set equal to 0.9
	F883	YH	R2 set equal to 90
	F887	YH	R2 set equal to 180
62	F88B	YH	R2 set equal to 0.6
6A	F88F	YH	R2 set equal to 1

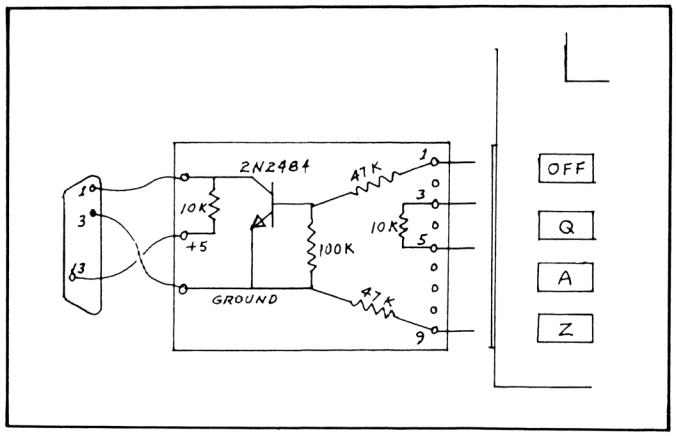
Note: In the last nine subroutines listed above, XH is loaded with #7A, and Flag C is set. The constants stored into R2 are obtained from a ROM table located FBE9 to FC9F.

THE RADIO SHACK PC-1 AS A DATA COLLECTING TERMINAL

One logical use for the Radio Shack TRS-80 PC-1 and Sharp PC-1211 is as a portable data entry device. The units have a keyboard, a readout, a reasonable amount of memory and the ability to retain information with the power off. Unfortunately, there is no built-in provision to transfer data to a larger computer!

This article describes a method of transferring data directly from a PC through a simple interface to a larger system. An accompanying drawing shows an interface that uses a single transistor. It can be connected between a PC-1/PC-1211 and a parallel input port of a larger computer that uses a CPU such as an 8080. The drawing illustrates the electronic schematic as well as the actual physical layout I have utilized. The pins that plug into the PC-1 are made out of number 24 tinned

Diagram *PC Data Transfer Interface*



copper wire and are about 3/8'ths inch in length. Construction is on a standard 0.1 by 0.1 inch perforated board, which provides the exact pin spacing needed. The transistor can be almost any small-signal NPN silicon device. The entire circuit may be enclosed in a small plastic box.

In my system, the +5 volts for the interface is obtained from my main computer, a NorthStar Horizon. I cut a trace on the mother board (that originally gounded pin 13 of the parallel input port) and routed +5 volts to it. The data line of the PC is connected to pin 1 of the Horizon's parallel port, which is bit 7, the most significant bit.

A machine language program is used in the Horizon to receive data from the PC-1. The first seven instructions clear the buffer where the data is to be stored. The main routine, labeled CHAR, brings in one character at a time and stores it in the buffer. It calls routine START to look for a start bit. If a start bit is found, routine NIBL is then used to bring in the first 4-bit nibble. Start contains a timing routine, TIMER, which provides a time delay of about five seconds. If a start bit is not detected within this time, the program jumps back to BASIC. This usually occurs only after all the data has been transferred. The DELAY routine provides a delay between bits.

The program is shown in assembled form as it would reside beginning at address 0. It could be relocated to suit other systems. Likewise, the memory area used as a buffer could be changed. To do this, the bytes at 01, 03, 04 and &0D (hexadecimal) would need to be altered. The buffer size I use consists of seven 256-byte blocks.

The format for each line of data is a start character (&8E for PRINT data and &8D for program statements), followed by the data and then 00 as a terminator. The program processes all characters using the hexadecimal values listed by Norlin Rober in Issue 05 of *PCN*.

If one wishes to change the period of TIMER, the byte at address &37 can be changed from &0B to some other value. Each loop is equal to about one-half second. The time value, &9C, at byte &83 in the DELAY subroutine is not critical. It should not have to be altered unless the speed of a particular PC-1 is unusually fast or slow. Both of these timing values are for a 4-megahertz system. They will need to be changed, of course, if your main computer operates at some other speed.

The program shown is in 8080 assembly code. It should work with little or no change in any microprocessor system using a similar CPU. The NorthStar Horizon I use has a Z80.

The programs I use to transfer data from the input buffer to data base files, are written in NorthStar BASIC. The address, &2A04, at bytes 18 and 19 of the transfer routine is an entry point to NorthStar BASIC that will not clear the program or variables. Other systems will require a different address.

There are a few unused bytes between some routines. All NOPs (no-operation bytes), however, should be included for the program to work properly.

Inputting Data to the PC

Data can be entered into the PC in a number of ways. I prefer inputting the data as simulated program statements, using the PRO mode. The

Program *PC Transfer Routines in 8080 Assembler Code*

```
00 00  3E 08       MVI A,08          ;END OF BUFFER
00 02  21 00 01    LXI H 01 00       ;START OF BUFFER
00 05  36 00       CLEAR, MVI M,00   ;CLEAR MEM LOC
00 07  23          INX H             ;INCR HL PAIR
00 08  BC          CMP H             ;COMPARE H WITH ACC
00 09  C2 05 00    JNZ CLEAR         ;CLEAR NEXT BYTE
00 0C  26 01       MVI H 01          ;RESET BUFFER
00 0E  00          NOP
00 0F  00          NOP
00 10  16 00       CHAR, MVI D,00    ;CLEAR REG D
00 12  CD 36 00    CALL START        ;LOOK FOR START BIT
00 15  AF          XRA A             ;CLEAR ACC
00 16  B8          CMP B             ;COMPARE WITH B
00 17  CA 04 2A    JZ 2A04           ;BACK TO BASIC
00 1A  CD 5D 00    CALL NIBL         ;GET FIRST NIBBLE
00 1D  50          MOV D, B          ;STORE IN D
00 1E  CD 36 00    CALL START        ;LOOK FOR START BIT
00 21  CD 5D 00    CALL NIBL         ;GET SECOND NIBBLE
00 24  78          MOV A, B          ;MOVE NIBBLE TO ACC
00 25  0F          RRC               ;SHIFT TO FAR RIGHT
00 26  0F          RRC
00 27  0F          RRC
00 28  0F          RRC
00 29  E6 0F       ANI 0F            ;NEED 4 LSB ONLY
00 2B  82          ADD D             ;COMBINE BOTH NIBBLES
00 2C  77          MOV M, A          ;STORE CHAR IN BUFFER
00 2D  23          INX H             ;INCR HL PAIR
00 2E  C3 10 00    JMP CHAR          ;GET NEXT CHAR

00 36  060B        START, MVI B, 0B  ;TIME CONSTANT
00 38  0E FF       MVI C, FF         ;TIME CONSTANT
00 3A  AF          XRA A             ;CLEAR ACC
00 3B  DB 00       LOOK, IN 0        ;READ PAR INPUT PORT
00 3D  A7          ANA A             ;SET FLAGS
00 3E  FA 4B 00    JM TIMER          ;IF NO START BIT
00 41  DB 00       IN 0              ;CHECK FOR TRUE BIT
00 43  A7          ANA A             ;SET FLAGS
00 44  FA 4B 00    JM TIMEA          ;IF NO START BIT
00 47  CD 82 00    CALL DELAY        ;HALF BIT DELAY
00 4A  C9          RET               ;BACK TO MAIN PROGRAM

00 4B  1D          TIMER, DCR E      ;KILL TIME
00 4C  C2 3B 00    JNZ LOOK          ;LOOK AGAIN
00 4F  0D          DCR C             ;KILL TIME
00 50  C2 3B 00    JNZ LOOK          ;LOOK AGAIN
00 53  05          DCR B             ;KILL TIME
00 54  C2 3B 00    JNZ LOOK          ;LOOK AGAIN
00 57  C9          RET               ;EXIT IF TIMED OUT

00 5D  06 00       NIBL, MVI B,00    ;CLEAR REG B
00 5F  0E 04       MVI C,04          ;SET BIT COUNTER
00 61  CD 82 00    BIT, CALL DELAY   ;SKIP FULL BIT
00 64  CD 82 00    CALL DELAY
00 67  AF          XRA A             ;CLEAR ACC
00 68  DB 00       IN 0              ;READ PAR PORT
00 6A  00          NOP
00 6B  E6 80       ANI 80            ;USE BIT 7 ONLY
00 6D  80          ADD B             ;ADD BITS IN REG B
00 6E  0D          DCR C             ;DECR BIT COUNTER
00 6F  CA 77 00    JZ END            ;TEST FOR 4 BITS
00 72  0F          RRC               ;SHIFT TO RIGHT
00 73  47          MOV B, A          ;STORE IN REG B
00 74  C3 61 00    JMP BIT           ;GET NEXT BIT
00 77  47          END, MOV B, A     ;STORE IN REG B
00 78  CD 82 00    CALL DELAY        ;MOVE PAST LAST BIT
00 7B  CD 82 00    CALL DELAY
00 7E  C9          RET               ;BACK TO MAIN PROGRAM

00 82  1E 9C       DELAY, MVI E,9C   ;NO.OF LOOPS FOR HALF BIT
00 84  1D          DEL, DCR E        ;KILL TIME
00 85  1C          INC E             ;KILL TIME
00 86  1D          DCR E             ;KILL TIME
00 87  C8          RZ                ;END OF DELAY
00 88  C3 84 00    JMP DEL           ;DELAY ANOTHER LOOP
```

LIST command is then used when transferring the data to my desktop computer. I restrict the use of line numbers to the range 10 to 99 to provide a constant line format. A decimal point is used as a separator between a line number and data and between fields of data.

Other ways to enter data are as PC BASIC strings. However, this method requires a short program in PC BASIC. Also, PC-1/PC-1211 strings are limited to seven characters. This complicates the program if several strings must be used for each record.

Data can also be stored as BASIC variables. This, too, requires a program. One advantage of this method is that the data can be preprocessed, if desired, with only the results stored for later transfer to the main computer. A disadvantage is that variables are transferred in the same format as they normally appear on the PC-1 display. Therefore a considerable number of space characters may also be processed.

If lines of data are limited to 20 characters each, the PC-1/PC-1211 will hold about 70 lines.

After the information has been collected and keyed into the PC, the procedure for passing data to a larger computer is as follows: Connect the pocket computer and interface to the larger system. Power up the large system and load the required programs. Enter the necessary commands to start the transferring program *except for the last key to be pressed.* The PC is then powered up by pressing the ON key twice. It is then placed in the RUN or LIST configuration, as required, except for the final press of the ENTER key. After this, press the last key needed to start the program in the large computer and then immediately press the ENTER key on the PC. When the data has been processed, the PC will display the > prompt. (The reason for starting both computers at the same time is so that the large unit will receive a start bit within five seconds. If it does not, it will jump back to its BASIC executive.)

After the data has been transferred to my Horizon, I use the accompanying program written in NorthStar BASIC to convert each line of data in the input buffer to string format. It is necessary to add other statements to this routine (after line 1190), if you want to move the strings to a file, convert them to numeric variables, etc. The function EXAM in lines 110 and 1820 is similar to PEEK in other BASICs. The routine could also be used to transfer PC BASIC programs. However, I have not completed the routine for such use. If desired, all of the PC BASIC tokens could be converted in statements like those beginning at line 300.

This method for collecting data with a pocket computer and transferring them to the Horizon for further processing has been a great time saver. It is often faster to enter numeric data in the field directly on the PC's keyboard rather than use pencil and paper. The automation also means fewer errors. Perhaps the greatest advantage of the system is the fact that the information collected is immediately available for further processing. Final results of a research project can be quickly and easily ascertained using the powers of the desktop computer.

Thanks go to: *S. J. Toy, Route 3, Box 73, Chico, CA 95926,* for this informative description of the PC as a data input terminal.

Program *Converting Raw Data to Strings*

```
40 DIM T$(80),A$(1),B$(80),D$(80)
45 DIM M1$(80),L1$(80),P1$(80)
78 P1=0:REM  OUTPUT TO CRT
79 REM  *****  CONVERT DATA TO STRINGS
80 M1$="E.9876543210       /°-+=[])("
85 L1$="ZYXWVUTSRQPONMLKJIHGFEDCBA"
95 P1$=":;,  $ %#!?"
100 A1=0255:REM  STARTING ADDR OF BUFFER
102 D$="":REM  INITIALIZE FOR CONCATENATION
105 A1=A1+1
110 C1=EXAM(A1):REM  GET CHARACTER
115 IF C1=141 THEN 105:REM  IGNORE LINE START CHAR FOR LIST
120 IF C1=142 THEN 105:REM  IGNORE LINE START CHAR FOR PRINT
130 IF C1=0 THEN 1100:REM  LINE END CHAR
140 IF C1[76 AND C1]47 THEN 185:REM  MATH SYMBOLS AND NUMBERS
150 IF C1[107 AND C1]80 THEN 195:REM  LETTERS
160 IF C1[30 AND C1]18 THEN 205:REM  PUNCTUATION MARKS
162 IF C1=18 THEN 175:REM  QUOTATION MARK
165 IF C1=17 THEN 210:REM  SPACE CHAR
170 IF C1[106 THEN 300 ELSE 900:REM  BASIC TOKENS
175 T1$=CHR$(34):GOTO 1000:REM  QUOTATION MARK
185 T1$=M1$(76-C1,76-C1):GOTO 1000
195 T1$=L1$(107-C1,107-C1):GOTO 1000
205 T1$=P1$(30-C1,30-C1):GOTO 1000
210 T1$=" ":GOTO 1000:REM  SPACE CHAR
300 IF C1=193 THEN T1$=" PRINT "ELSE 310:GOTO 1000
310 IF C1=194 THEN T1$=" INPUT "ELSE 900:GOTO 1000
315 REM  ADD OTHER KEYWORDS OF PC BASIC IN SAME FORMAT
900 T1$=" ":GOTO 1000:REM  PRINT SPACE IF NO CONV CHAR
1000 D$=D$+T1$:GOTO 105:REM  ADD CHAR TO STRING
1100 REM  STRING CONVERSION COMPLETED
1190 !#P1,D$:REM  PRINT THE CONVERTED STRING
1810 A1=A1+1:REM  INCR CHAR LOCATION
1820 C1=EXAM(A1):REM  GET NEXT CHAR
1830 IF C1[141 OR C1]142 THEN 5000:REM  TEST FOR LINE START CHAR
1900 D$=B$:REM  CLEAR DATA STRING
1910 D$="":REM  INITIALIZE DATA STRING
2000 GOTO 105:REM  PROCESS ANOTHER LINE
5000 !:!" DATA ALL PROCESSED":!
```

FROM THE WATCH POCKET

A number of readers have asked me to comment further on the HP-75C that I reviewed several months ago. There is insufficient space for extensive comment in this issue, but if interest continues I shall plan a column later. Let me just say that after several hundred operating hours with the HP-75C, my respect and admiration for the machine continues to grow. It is a very pleasurable machine to utilize. At this point, I know that the feature I enjoy the most is its file handling capabilities. That is, the ability to instantly switch between and access many different files within its memory. It is a powerful unit and I continue to believe that it has set a standard against which other entries to the field will have to be compared. Yes, it is a relatively expensive unit. The challenge for others is whether they can begin to match it and cause a lowering of its price? In summary, at this point, I am far more impressed with, say, HP's entry to the portable computing market as evidenced by their HP-75C, than I was a few years ago when they entered the personal computing market with their HP-85. They could, in fact, improve the HP-85 by adding some of the capabilities of the 75!

The Year in Review

There was nowhere near the momentum or sales that could well have been expected in the pocket computer market. Here are my year-end "guess-estimates" of U.S. sales for some of the leading PCs.

Sharp PC-1211/Radio Shack PC-1 (combined) 1982 sales of 100,000. Down from perhaps 175,000 in 1981. This model is rapidly being outdated, they are being discounted heavily by just about everyone except Radio Shack. I doubt that you will be able to even buy one by the end of 1983.

Sharp PC-1500/Radio Shack PC-2 does not seem to be as doing as well as might have been expected. Perhaps 40,000 - 50,000 units sold so far. Many reasons, in my opinion, for the relative softness: Initial poor marketing, followed by better (but too little and too late). A person who views the powerful one-minute television ads for a Timex-Sinclair at $100.00 has no comprehension of how a PC-1500/PC-2 compares when observing a single-page magazine ad. The essential key asset of a PC (its battery powered portability) is never properly highlighted nor the ramifications explained. The distribution channels were not properly chosen for this type of product. There seems little motivation for R.S. Computer Centers to push this item when a salesman can make a lot more commission promoting a desktop unit. Sharp apparently has missed out on a once-in-a-lifetime opportunity to be a major force in the U.S. market. In fact, I tend to think they never fully understood the opportunities that they had here. It seems obvious that they could have done much better. There are a lot of competitors coming along that will probably be able to avoid their mistakes. If Sharp does not develop a better understanding of the U.S. market in the near future, they may quickly lose their status as being a front-runner in this volatile field.

Panasonic/Quasar is apparently not pressing for the consumer market. It is hard to get a handle on their success in the industrial arena. I would be surprised if they did more than about 30,000 units overall (remember, I am considering only the U.S. market). I hear rumors of some sizable sales (a few hundred units) to firms that will develop special application modules. But, the relative price and lack of certain basic amenities seems to have kept them out of the so-called "popular" market. I don't expect much out of them in 1983.

Casio apparently never even knew what hit them. The FX-702P appears to have lacked a few fundamental features desired by PC users. Despite its touted special function capabilities, it never appeared to gain any marketing momentum. (One thing that probably hurt them was their inability to come up with the promised plug-in modules. A number of people told me they were waiting to see what those provided. When they never appeared, the interest in the product died.) I figure they would have been lucky to sell 20,000 units in 1982. I do not expect to see much in the way of wonders from Casio in 1983.

It really seems that PC manufacturers, as a group, are being creamed by the marketing clout of those such as Commodore, Atari and Texas Instruments. These latter firms are convincing newcomers that the way to join the computer revolution is through an attachment to their TV. Play games, they say, and do it all in living color. The Timex-Sinclair unit is reported as being manufactured at a rate approaching 75,000 per month and the Commodore VIC-20 at around 40,000 per month. Look at what the PC manufacturers have missed!

Where the Market is Headed

People definitely want personal, portable computing power. They also want convenience. PC makers had better plan on offering at least two features if they want to stay in the game: The first is improved display capability (multiple-lines). The second is a comfortable keyboard. A keyboard such as that on a PC-1500, which was acceptable to the pioneering 1982 buyers, will not be tolerated by the mass-market consumers. They have better sense!

Two companies are already showing the trend. Hewlett-Packard, with its well-designed HP-75 (though they could soften up their keyboard), and Epson with their 4-pound HX-20.

I would tend to think that Epson, with their several years of experience marketing their popular line of printers in the U.S., would be in a position to exploit a wide-open opportunity. However, they too may blow it. I have learned that they have recently been shaking up and re-arranging their entire U.S. distribution system. Apparently they are going to try to keep the HX-20 out of the hands of mail order suppliers, a la Apple Computer Company's course. And, they may try to restrict access to technical information a la Sharp. Adopting either of these philosophies could hurt their penetration and long term sales potential, just as it has done to their predecessors.

PCs and Telecomputing: Not Good Enough – Yet!

A lot of companies appear to be trying to sell the idea that a PC is just what one needs to do telecomputing. (That is, have the PC act as a remote terminal for a large computer.) They are coming out with modems, RS-232C interfaces and "terminal software" (you could sure read that with more than one meaning) so that you can hook a PC to a telephone and talk to telecomputing networks such as The Source and Compuserve. After trying it out for awhile, I am thoroughly disappointed.

There are several problems. The two most poignant are: The PC keyboards, for the most part, are too small to permit touch-typing and so you spend too much time inputting commands and/or data. And, the displays are too small to permit comfortable viewing of the volumes of information that one has access to. I defy anyone to spend an hour trying to use Compuserve using a one-line PC display and come away without either having a severe headache or the feeling that your eyeballs had been through an experience for which they were not designed.

Oh yes, the idea is great. It would be nice to be able to access some of those data bases at anytime and from any location. It is the present implementation of the idea that is sorely lacking.

Here is what is needed to rectify the situation: A display that will allow at least 10 to 15 seconds of transmitted data to appear on the screen at one time. That is just enough time for you to make a rapid scan and consider whether the information has lasting value to you personally. (This could be achieved by using a TV or video interface. Some units, such as Panasonic's HHC and HP's 75C have this capability.) A printer unit that will allow you to make a hardcopy of important information. (Again, a number of PCs can be connected to printers. Some of these, however, may not operate fast enough for typical telecomputing applications.) And, most importantly (which no manufacturer that I know of has yet provided), a control arrangement that would allow you to turn the printer on and off as needed. It is important to realize that the printer must be capable of picking up what is currently in the PC's display buffer. Thus, if you were scanning data and you suddenly realized that you wanted to retain the information you were viewing, you would have several seconds in which to activate the printer and capture the relevant data!

These are all simple concepts, easy to implement with today's technology. The first company to be able to supply such a unit could really sock it to those firms that haven't figured on the importance of human factors engineering.

Thank you for supporting *PCN* during 1982. Best wishes to everyone for prosperity during 1983 and beyond. – *Nat Wadsworth, Editor*

1982 INDEX TO ARTICLES AND PROGRAMS

ENTERTAINMENT PROGRAMS

Auersbacher, Emerich:
Electronic Card Deck . . . . . 11 - 8
Heidbrink, Gary:
American Roulette . . . . . 17 - 7
Motto, David:
Animated Runner . . . . . 14 - 3
LCD Graphics for the PC-1500 . . . . . 14 - 5
Shifty Words . . . . . 12 - 8
Peterson, Brian:
Generating the Chromatic Scale . . . . . 15 - 1
Music on the PC-1500 . . . . . 16 - 5

GENERAL APPLICATIONS

Auersbacher, Emerich:
Amortization . . . . . 13 - 3
Calendar . . . . . 12 - 4
Coding/Decoding Messages . . . . . 14 - 1
Phases of the Moon . . . . . 14 - 8
Stopwatch . . . . . 19 - 6
Weather Forecasting . . . . . 20 - 2
Griffiths, Herb:
Printer's Helper . . . . . 16 - 6
Motto, David:
Time . . . . . 14 - 3
Nath, H.:
Accounts Receivable . . . . . 12 - 6
Peterson, Brian:
Memo Printer . . . . . 19 - 5
Robertson, Gene:
Aviator . . . . . 11 - 6
Saam, R. J.:
Tape Library System . . . . . 17 - 2
Slaughter, Ken:
Determine Tab Settings . . . . . 13 - 2
Supermarket Companion . . . . . 11 - 4
Sparkman, James K.:
Relative Humidity & Dew Point . . . . . 12 - 4
Stevens, Geurri F.:
Tax Assessment . . . . . 17 - 4
Stonebrook, Kendal B.:
Indirect File Names on Tape . . . . . 11 - 1
Toy, S. J.:
PC-1 as a Data Collecting Terminal . . . . . 20 - 5

OPERATING INFORMATION

Auersbacher, Emerich:
Tips & Tricks . . . . . 13 - 8
Battan, Howard R.:
Index for the PC-1500 Manual . . . . . 16 - 8

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January – December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

☐ 1982 Regular Subscriber (Issues 11 – 20). $30.00 in U.S. (U.S. $36.00 to Canada. U.S. $45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 – 30). $60.00 in U.S. (U.S. $72.00 to Canada. U.S. $90.00 elsewhere.)

☐ 1983 Regular Subscriber (Issues 21 – 30). $36.00 in U.S. (U.S. $42.00 to Canada. U.S. $50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: ___ Zip: _____

MC/VISA #: _____ Expires: _____

Signature: _____

P.O. Box 232, Seymour, CT 06483

Chamier, A. D.:
X—Y Plotter Interface . . . . . 13 - 6
Cox, Thomas S.
Converting Statements . . . . . 19 - 1
Error Codes for the PC-1500 . . . . . 16 - 5
Status Report . . . . . 16 - 5
Fergus, George:
Put Softkeys to Work . . . . . 14 - 5
Speed Insertion of Statements . . . . . 11 - 8
Hart, Rich:
Op-Aids for the PC-1500/PC-2 . . . . . 19 - 7
Miller, Ron:
PC Interface Signals . . . . . 13 - 5
Motto, David:
Comparison of PC Memories . . . . . 16 - 4
Upper Case Conversion . . . . . 16 - 1
Rober, Norlin
Exploring the PC-1/PC-1211 . . . . . 13 - 4
Getting Started With Machine Language . . . . . 20 - 3
Hexadecimal Memory Dump . . . . . 15 - 3
PC-1 Instruction Execution Times . . . . . 11 - 2
PC-1500/PC-2 Disassembler . . . . . SE - 4
PC-1500/PC-2 Machine Language Codes . . . . . SE - 1
Peeking in the PC-1500 . . . . . 15 - 3
Rober Mnemonics Update . . . . . 18 - 5
Understanding the PC-1500 Part 1 . . . . . 15 - 2
Understanding the PC-1500 Part 2 . . . . . 16 - 2
Understanding the PC-1500 Part 3 . . . . . 19 - 2
Use of Memory in the Radio Shack PC-1 . . . . . 13 - 4
Sherwin, Milt:
PC-1500/PC-2 Renumbering Program . . . . . 18 - 5
Renumbering Revisited . . . . . 19 - 3
Staff, PCN:
Converting Programs for the FX-702P . . . . . 16 - 3
Softkeys Under Program Control . . . . . 14 - 8
Updates to the PC-1500 Manual . . . . . 16 - 6

PRODUCT ANNOUNCEMENTS

Staff, PCN:
Epson Announces HX-20 Portable . . . . . 20 - 1
FX-700P Cracks $100.00 Barrier . . . . . 18 - 1
HHC Service on Compuserve . . . . . 19 - 1
HP Announces Programmer's Aid . . . . . 16 - 1
HP Introduces Portable Computer . . . . . 17 - 1
Protection For Your PC . . . . . 17 - 8
Radio Shack Announces New PC . . . . . 12 - 2
Radio Shack Four-Color Printer/Plotter . . . . . 12 - 2
Sharp Prepares New PC-1500 . . . . . 11 - 1
Sharp Announces PC-1500 . . . . . 12 - 1
Transfer Data from PC . . . . . 17 - 8

PRODUCT REVIEWS

Motto, David:
Panasonic HHC . . . . . 14 - 2
Sharp PC-1500 . . . . . 14 - 3
Rober, Norlin:
Electrical Engineering I . . . . . 14 - 5
Staff, PCN:
Sharp PC-1500 . . . . . 13 - 1
Wadsworth, Nat:
Aviation Program . . . . . 11 - 2
HP-75: Formidable Companion . . . . . 18 - 2
Problem Solving on the PC-1 . . . . . 14 - 6
Sharp CE-150 Printer/Plotter . . . . . 14 - 4
Telecomputing with a HHC . . . . . 16 - 7
99 Tips & Tricks for the New PCs . . . . . 20 - 2

SCIENTIFIC/ENGINEERING PROGRAMS

Auersbacher, Emerich:
Any Base Conversion . . . . . 11 - 7
Chamier, A. D.:
Fast-Fourier-Transform . . . . . 13 - 3
Cox, Thomas S.:
Curve Fitting . . . . . 15 - 4
Decimal to Bases 2 – 16 Conversion . . . . . 17 - 6
Librach, Hank:
Low Pass Filter Design . . . . . 12 - 5
Meredith, Paul T.:
Interpoloation Package . . . . . 12 - 3
Rober, Norlin:
Linear Regression Analysis . . . . . 11 - 6
Polynomial Arithmetic . . . . . 12 - 5
RPN Calculator for PC-1/PC-1211 . . . . . 18 - 7
Triangles . . . . . 11 - 5
Shuman, David:
Electronic Filter Design on the FX-702P . . . . . 14 - 7